BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF MECHANICAL ENGINEERING
FAKULTA STROJNÍHO INŽENÝRSTVÍ

INSTITUTE OF MATHEMATICS
ÚSTAV MATEMATIKY

MODELS AND METHODS FOR ROUTING PROBLEMS IN LOGISTICS
MODELY A METODY PRO SVOZOVÉ PROBLÉMY V LOGISTICE

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR Izza Hasanul Muna
AUTOR PRÁCE

SUPERVISOR RNDr. Pavel Popela, Ph.D.
VEDOUCÍ PRÁCE

BRNO 2019

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF MECHANICAL ENGINEERING

Master's Thesis Assignment

Institut:	Institute of Mathematics
Student:	**Izza Hasanul Muna**
Degree programm:	Applied Sciences in Engineering
Branch:	Mathematical Engineering
Supervisor:	**RNDr. Pavel Popela, Ph.D.**
Academic year:	2018/19

As provided for by the Act No. 111/98 Coll. on higher education institutions and the BUT Study and Examination Regulations, the director of the Institute hereby assigns the following topic of Master's Thesis:

Models and methods for routing problems in logistics

Brief description:

The student will deal with mathematical modeling for various logistic and distribution problems involving routing decisions. The formulated models will be based on linear and integer programming and mixture of them. For the formulated problems, he will discuss and suggest a solution methods (either exact or heuristic). Since the demand is very often hard to predict, uncertainty will also be included/discussed. The thesis will be written in cooperation with Dr. Dušan Hrabec who will be involved as an external advisor.

Master's Thesis goals:

An overview of selected models and methods for decision making in logistic will be presented.
Mathematical models for selected logistic problems will be designed and their properties will be studied.
The solution algorithms will be discussed and developed by modification of existing ones.
The software implementation of models and sloution methods will realized.
The obtained test results will by analysed and discussed.

Recommended bibliography:

BAZARAA, Mokhtar S., JARVIS, John J. and SHERALI, Hanif D. Linear programming and network flows. 4th ed. Hoboken, N.J.: John Wiley & Sons. 2010. ISBN 978-0-470-46272-0.

BIRGE, John R. and LOUVEAUX, François. Introduction to Stochastic Programming. Springer Verlag. 1997. ISBN: 978-1-4614-0236-7.

CHRISTOFIDES, Nicos. Graph Theory - an Algorithmic Approach. Academic Press. 1975. ISBN 0-12-174350-0.

GHIANI, Gianpaolo, LAPORTE, Gilbert and MUSMANNO, Roberto. Introduction to logistics systems planning and control. Hoboken, NJ, USA: J. Wiley. 2004. ISBN 0-470-84917-7.

KALL, Peter and WALLACE, Stein W. Stochastic Programming. New York: John Wiley & Sons. 1993. ISBN 978-0471951582.

WILLIAMS, H. Paul. Model building in mathematical programming. 5th ed. Hoboken, N.J.: John Wiley & Sons. 2013. ISBN 978-1-118-44333-0.

WOLSEY, Laurence A. Integer programming. New York: John Wiley & Sons. 1998. ISBN 978-0-471-28366-9.

Students are required to submit the thesis within the deadlines stated in the schedule of the academic year 2018/19.

In Brno, 2. 11. 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ
DĚKANÁT
L. S.

prof. RNDr. Josef Šlapal, CSc.
Director of the Institute

doc. Ing. Jaroslav Katolický, Ph.D.
FME dean

Abstract

The thesis focuses on how to optimize vehicle routes for distributing logistics. This vehicle route optimization is known as a vehicle routing problem (VRP). The VRP has been extended in numerous directions for instance by some variations that can be combined. One of the extension forms of VRP is a capacitated VRP with stochastics demands (CVRPSD), where the vehicle capacity limit has a non-zero probability of being violated on any route. So, a failure to satisfy the amount of demand can appear. A strategy is required for updating the routes in case of such an event. This strategy is called as recourse action in the thesis.

The main objective of the research is how to design the model of CVRPSD and find the optimal solution. The EEV (*Expected Effective Value*) and FCM (*Fuzzy C-Means*) – TSP (*Travelling Salesman Problem*) approaches are described and used to solve CVRPSD. Results have confirmed that the EEV approach has given a better performance than FCM-TSP for solving CVRPSD in small instances. But EEV has disadvantage, that the EEV is not capable to solve big instances in an acceptable running time because of complexity of the problem. In the real situation, the FCM –TSP approach is more suitable for implementations than the EEV because the FCM – TSP can find the solution in a shorter time. The disadvantage of this algorithm is that the computational time depends on the number of customers in a cluster.

Keywords

logistics, graphs, optimization, routing problem, vehicle routing problem with uncertain demands, fuzzy c-means (FCM) – TSP.

Bibliographic citation

MUNA, I. H. *Modely a metody pro svozové problémy v logistice.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2019. 44 s. Vedoucí diplomové práce RNDr. Pavel Popela, Ph.D.

Declaration

I declare that the presented thesis is the result of my own work under the guidance of my supervisor and I cited all literature and electronic sources what I used during the research.

In Brno, 26 April 2019

Izza Hasanul Muna

ACKNOWLEDGEMENT

I would like to express gratitude to my supervisor RNDr. Pavel Popela, Ph.D. and my external advisor Dr. Dušan Hrabec for supporting, consultations, patience and valuable advice to my thesis. My thanks also belong to my colleagues from InterMaths programme. Finally, I am most grateful to my parents, my entire family and Elok Mutiara Rakhmawati who supported me every time.

Contents

Chapter 1 ..... **9**

1.1. Introduction 9

1.2. Thesis Structure 10

Chapter 2 ..... **11**

2.1. Graph Theory 11

2.2. Travelling Salesman Problem (TSP) 13

2.3. Vehicle Routing Problem (VRP) 15

2.4. Capacitated-VRP (CVRP) 18

2.5. CVRP with Stochastics Demands (CVRPSD) 19

Chapter 3 ..... **20**

3.1. Recourse Policy 20

3.2. EEV (*Expected Effective Value*) for CVRPSD 20

3.3.Fuzzy C-Means (FCM) – TSP Method for finding Heuristic Solution of CVRPSD 25

Chapter 4 ..... **30**

4.1 Computational Results and Discussion 30

Chapter 5 ..... **36**

References **37**

List of Figures **39**

List of Tables **40**

Abbreviation list **41**

Appendix A **42**

Appendix B **44**

Appendix C **46**

Chapter 1
Introduction & Thesis Structure

1.1. Introduction

In Industry, logistics is one of the most important aspects that needs to be considered by companies. Although many companies focus on the design and production of their products and services to best meet customer needs, if those products cannot reach customers, the business will fail and cannot survive. Moreover, if a company can manage logistics effectively and efficiently, this means the company can optimize costs, energy and time such that it will affect to the profits. In other words, companies should concern in a distribution problem, how to distribute products and organize logistics properly such that a high profit can be achieved.

Practically, there are several obstacles that must be satisfied in the distribution process, such as types of demands, vehicle capacities, delivery deadline, average speed that can be reached on certain paths and times, different locations of consumers, etc. Thus, in order to deal with this problem, companies need to optimize vehicle routes so that all of obstacles and restrictions can be satisfied. This vehicle route optimization is known as vehicle routing problem (VRP). The VRPs are related to combinatorial optimization and integer programming introduced by Dantzig and Ramser more than 50 years ago (1959) [1]. VRP belongs to the category of NP hard problems that can be exactly solved only for small instances of the problem. The VRP aims to deliver goods to consumers with the optimum route and minimize the number of vehicles used to get in and out of the depot [2].

In 1964, Clarke and Wright [1] extended VRP and obtained Capacitated-VRP (CVRP) and proposed the first heuristics for this problem. In the CVRP, we

have given a depot, a set of n customers, a set of m vehicles with capacity K and every customer has a demand d_i. The task in the CVRP is to construct vehicle routes such that all of customers are served exactly once and no vehicle visits a set of customers whose total demand exceeds K [3]. Regarding this, the research is focused on CVRP with stochastics demands (CVRPSD), where the demands are unknown until vehicles arrive at the customer. This problem will be described in more detail in the following sections.

1.2. Thesis Structure

This thesis is organized as follows:

1. Chapter 1 gives a general introduction to the CVRPSD.
2. Chapter 2 is focused on preliminary material that we need to understand what is VRP and its extensions. In particular, literatures resources on the graph theory, TSP, VRP, CVRP and the CVRPSD are studied for a deeper understanding of the problem studied.
3. Chapter 3 begins the discussion on the methodology how to solve CVRPSD. EEV (*Expected Effective Value*) and Fuzzy C-Means (FCM) – TSP methods are adopted to generate solution for CVRPSD. This chapter also discusses the mathematical model of CVRPSD and its recourse action.
4. Chapter 4 presents 2 methods that we have chosen in order to solve CVRPSD and discuss their effectiveness and applicability. We test the methods on a set of instances. Then we discuss the behavior of each method and evaluate the result by changing the number of scenarios.
5. For the last chapter, conclusions are drawn and recommendation of future research on this problem is given.

Chapter 2
Preliminaries

2.1. Graph Theory

2.1.1. Graph Data Structure

A graph G is a collection of points or vertices $v_1, v_2, v_3, \dots, v_n$ (denoted by the set V) and a collection of edges $e_1, e_2, e_3, \dots, e_n$ (denoted by the set E) joining all or some of these points. Then graph G is fully described by the doublet (V, E), see [4] for more details. Figure 2.1 is an example of $G_1(V, E)$ with $V = \{v_1, v_2, v_3, v_4\}$ and $E = \{e_1, e_2, e_3, e_4\}$. Since the lines in graph G_1 have no orientation, they are called edges (*links*) and G_1 is an *undirected* graph. If the edges in E have directions as is shown in the graph G_2(Figure 2.2), then the resulting graph is called a *directed* graph. In the *directed* graph, $(v_j, v_k) \neq (v_k, v_j)$ and edges are also called arcs.

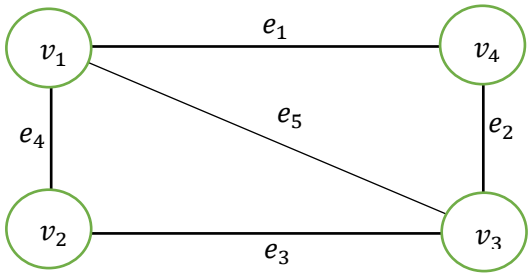

Figure 2.1: Graph G_1

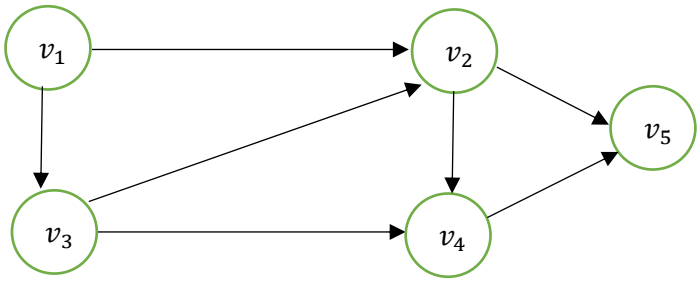

Figure 2.2: Graph G_2

2.1.2. Connectivity

2.1.2.1. Walk, trail, path, circuit and cycle

This part will explain definition of a walk, trail, path, circuit and cycle, see [4] for their precise formulation.

(a) A walk is defined as a sequence of alternating vertices and edges consecutive elements of which are incident, that begins and ends with a vertex. In graph G_1, $w_1 = (v_1, e_1, v_4, e_2, v_3)$ is a walk.

(b) A trail is the walk without repeated edges. $w_1 = (v_1, e_1, v_4, e_2, v_3)$ is a trail in graph G_2.

(c) A path is defined as a sequence of distinctive vertices connected by edges. In a *directed* graph, path is any sequence of arcs where the final vertex of one arc is the initial vertex of the next arc. A sequence of arcs v_1, v_3, v_2, v_5 in Figure 2.2 is a path.

(d) A Circuit is a closed trail, meaning we start and end at the same vertex. A circuit has no repeated edges but may have repeated vertices.

(e) A cycle is defined as a closed trail where no other vertices are repeated apart from the start/end vertex. $w_2 = (v_1, e_1, v_4, e_2, v_3, e_3, v_2, e_4, v_1)$ is an example of a cycle in graph G_1. If the cycle passes through all the vertices of a graph, then it is called as a *Hamiltonian* cycle and a graph containing *Hamiltonian* cycle is so called as *Hamiltonian* graph [4].

2.1.2.2. Connectivity of graph

One of the important things in graph theory is regarding connectivity. Connectivity in a graph is divided into two concepts, namely vertex connectivity and edge-connectivity. Vertex-connectivity is given by the minimum number of vertices whose removal makes graph either disconnected or reduces graph into a trivial graph. Edge-connectivity is the minimum number of edges whose removal makes a graph to be disconnected [5].

More precisely, a graph is said to be connected if there is a path between every pair of vertices. It means that from every vertex to any other vertex, there should be some path to traverse. That is called the connectivity of a graph. Conversely, a graph with multiple disconnected vertices and edges is said to be disconnected. Figure 2.3 shows an example of connected graph and disconnected graph.

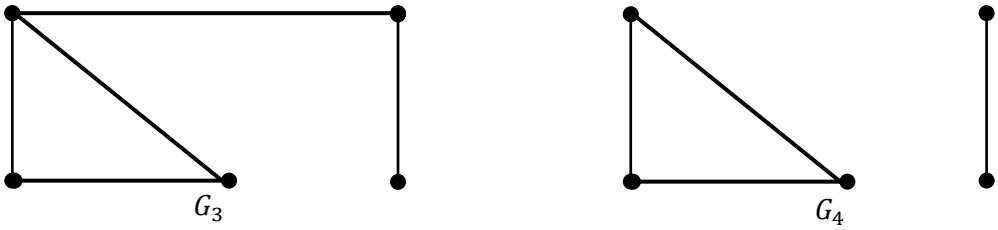

Figure 2.3: Connected graph G_3 and disconnected graph G_4

2.1.3. Weighted Graph

A weighted graph G is a graph in which a number (the weight) is assigned to each edge. These weights might represent for example costs, lengths or capacities, depending on the problem. The example of weighted graph is given in Figure 2.4.

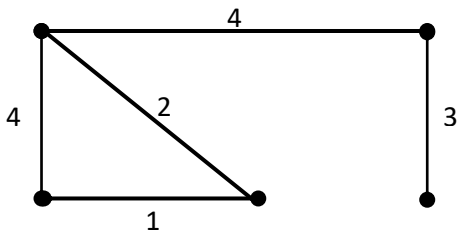

Figure 2.4: Weighted graph

2.2. Travelling Salesman Problem (TSP)

The TSP is a classical problem that tries to find the shortest path that a salesman goes through. The problem describes a salesman who wants to travel between *N* cities and he has to visit each one of the cities starting from a certain one, and finishes where he was at first. In formal way, TSP is defined as an indirect

graph where the edges have weight and will be searched for Hamilton circles with a minimum total weight [6].

The TSP can be formulated as an integer linear program. There are two notable formulations, defined by Miller-Tucker-Zemlin (MTZ) and the Dantzig-Fulkerson-Johnson (DFJ). Although MTZ is weaker than DFJ [7], in this chapter, only MTZ formulation will be given since this formula is still useful in certain further discussed settings. Firstly, define the binary variables

$$x_{ij} = \begin{cases} 1, & \text{if arc(i,j) is used on the tour} \\ 0, & \text{otherwise} \end{cases}$$

for $i = 1,..n$. Let u_i be a dummy variable and take c_{ij} to be the distance from city i to city j. Then, TSP can be written as follows:

$$z = \min\left\{\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}\right\} \tag{1}$$

Subject to:

$$\sum_{i=1,i\neq j}^{n} x_{ij} = 1, j = 1,..n \tag{2}$$

$$\sum_{j=1,i\neq j}^{n} x_{ij} = 1, i = 1,..n \tag{3}$$

$$u_i - u_j + nx_{ij} \leq n-1, \quad 2 \leq i \neq j \leq n \tag{4}$$

$$x_{ij} \in \{0,1\}, \qquad i,j = 1..n$$

$$0 \leq u_i \leq n-1, \; 2 \leq i \leq n$$

The first constraint (2) requires that each city is arrived at from exactly one other city, while (3) requires that from each city there is a departure to exactly one other city. The last constraint (4) enforce that there is only a single tour covering all cities, and not two or more disjointed tours that only collectively cover all cities. In other words, the last constraints are used to exclude sub-tours.

In order to make easy to understand TSP, consider the set of cities in figure 2.5. There are 5 cities, denoted by A, B, C, D and E. The problem lies in finding a minimal path passing from all vertices once. For instances the path 1 {A, B, C, D, E, A} and the path 2 {A, B, C, E, D, A} pass all the vertices. The better solution for this problem is path 1 since path 1 has a total length of 24 and path 2 has a total length of 31.

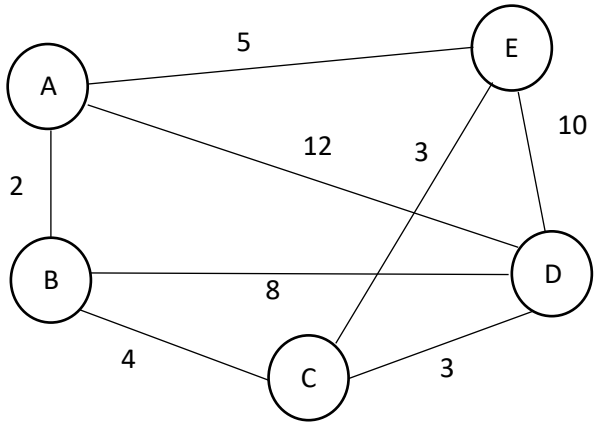

Figure 2.5: TSP on a graph with weights on its edges

2.3. Vehicle Routing Problem (VRP)

A VRP is a generalization of the Traveling Salesman Problem. The VRP consist of determining the routes to be used by a fleet of vehicles to serve a set of users. In mathematical way, VRP can be defined on a mixed graph $G = (V, A, E)$, where V is a set of vertices, A is a set of arcs and E is a set of edges. A vertex 1 represents the depot at which *m* vehicles are based. In this graph representation, arcs and edges correspond to road segments, and vertices correspond to road intersections [4].

There are many methods to solve the VRP, but the two most widely used methods are the exact and heuristic methods. Table 2.1 shows the most extensively used exact and heuristic procedures in solving the VRP [8]. Exact methods explore a whole state space and try to find the best solution. If the number of customers is small, exact methods can be used to optimally solve the

problem, but because the problem complexity is very high for bigger instances, exact approach becomes inefficient since it is time consuming [6].

Table 2.1: Some methods to solve VRP

Type of Approach	Solution Procedure
Exact methods	Integer Programming
	Mixed Integer Programming
	Branch and Bound
	Branch and Cut
	Pricing
Heuristics	Insertion algorithm
	Savings algorithm
	Clustering algorithm

The second set of methods, contains heuristics that are problem-dependent techniques. They are usually adapted for the problem and they often take advantage of problem specifics, but as they are often too greedy, they usually get trapped in a local optimum [9]. In the real problems that involve large amounts of data input, heuristics are more applicable than exact methods since they are faster to produce the solution than exact methods. But the solution can be produced by heuristics has usually moderate quality.

2.3.1. Formula of VRP

Based on double-index integer linear programming formulation [10], the formula of basic VRP is described as follows:

Define the binary variables $x_{ij} = \begin{cases} 1, & \text{if arc(i,j) used on the tour} \\ 0, & \text{otherwise} \end{cases}$

$$z = \min\left\{\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}\right\} \tag{5}$$

Subject to:

$$\sum_{j=2}^{n} x_{1j} = m \tag{6}$$

$$\sum_{j=2}^{n} x_{j1} = m \tag{7}$$

$$\sum_{i=1}^{n} x_{ij} = 1, j = 2, .. n \tag{8}$$

$$\sum_{j=1}^{n} x_{ij} = 1, i = 2, .. n \tag{9}$$

$$\sum_{i \epsilon S}^{n} \sum_{j \epsilon S}^{n} x_{ij} \leq |S| - 1, \ \forall S \subseteq V \ \backslash \{v_1\} \tag{10}$$

$x_{ij} \in \{0,1\}, \forall (i,j) \in A,$

where m is the number of vehicles and n is the number of customers.

Constraints (8) and (9) are the usual assignment constraints, (6) and (7) ensure that exactly *m* vehicles depart from and return back to node 1 (the depot). Constraints (10) are used to avoid sub-tours in the solution, i.e. cycling routes that do not passthrough the depot. The objective function is defined by (5) and imposes that the total travel cost of the routes is minimized.

Typically, there are some variations of VRP as shown in figure 2.6. As can be seen in the figure, new variant is usually created by adding some constraint, which is taken from practical application (e.g. vehicle capacity, customers time windows or multiple depots), or by combining existing variants. For more details, we advise the reader to see [6], [8] and [11].

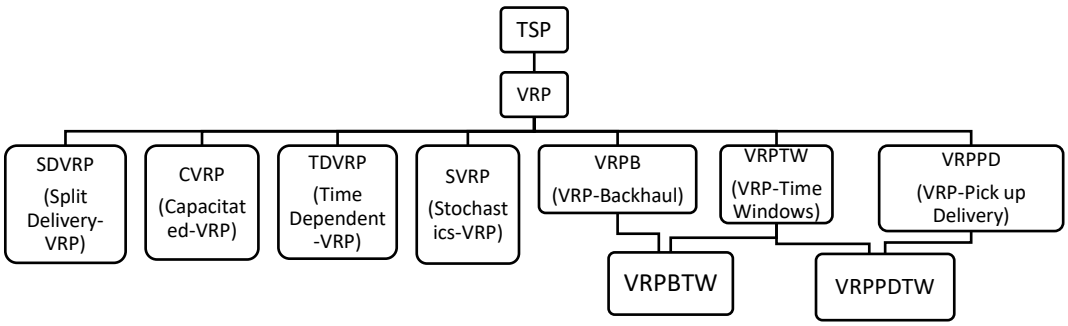

Figure 2.6: VRP variants

2.4. Capacitated-VRP (CVRP)

CVRP is a VRP with additional constraints on the capacities of the vehicles. The objective is to find a collection of routes of minimum total travel cost under the restrictions that each route begins and ends at the depot, each customer is serviced exactly once, and the total demand on any route does not exceed the vehicle capacity [12].

Basically, the formula of CVRP is almost the same with basic VRP, but CVRP has additional constraints on the capacities of the vehicles. Thus, based on [13], CVRP can be formulated as following:

Define the binary variables $x_{ij} = \begin{cases} 1, & \text{if arc(i,j) is used on the tour} \\ 0, & \text{otherwise} \end{cases}$

$$z = \min\left\{\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}\right\} \tag{11}$$

Subject to:

$$\sum_{j=2}^{n} x_{1j} = m \tag{12}$$

$$\sum_{j=2}^{n} x_{j1} = m \tag{13}$$

$$\sum_{i=1}^{n} x_{ij} = 1, j = 2,..n \tag{14}$$

$$\sum_{j=1}^{n} x_{ij} = 1, i = 2,..n \tag{15}$$

$$u_j - u_i + K(1 - x_{ij}) \geq d_j, i, j \in V\backslash\{1\} \tag{16}$$

$$x_{ij} \in \{0,1\}, \forall (i,j) \in A$$

where u_j is dummy variable, K is total capacity and d_j is demand at each node j. The constraints $(12) - (15)$ are VRP constraints, whereas (16) impose both the capacity and connectivity of the feasible routes.

2.5. CVRP with Stochastics Demands (CVRPSD)

The CVRP with Stochastic Demands (CVRPSD) is a modified version of CVRP, where customers have stochastic demands such that, in general, the vehicle capacity limit has a non-zero probability of being violated on any route. It is a stochastic integer linear program, which can be modeled by two main approaches: *Chance Constrained Programming* (CCP) and *Stochastic Programming with Recourse* (SPR) [14]. In this section, only CVRPSD modeled by SPR is provided since it has a wider range of applications than CCP models.

In the CVRPSD, the total actual demand on a route may exceed the vehicle capacity since the demands are unknown until vehicle arrives at customer [15]. As a consequence, a vehicle might not be able to load all of the actual customer demands on any given route having more than one customer. The CVRPSD by SPR approach deals with this issue by permitting recourse actions. These actions lead to extra costs for routes, which we call penalty costs, and it is generally possible to compute the expected penalty cost of a route induced by the stochastic demands. Since the particular strategy affects the expected cost of a given route, the strategy must be known at the time of planning [16].

In general, CVRPSD can be defined in many different ways depending on the recourse policy used. In the next sections, a methodology for solving this problem and its model will be given.

Chapter 3
Methodology

3.1. Recourse Policy

In the thesis, we will consider recourse policies under which each vehicle serves customer in its route according to so called “splitable demand”. It means the amount of demand of customer j can be partitioned. So, when the expected demand in customer j is strictly greater than vehicle capacity, the vehicle will leave the remaining amount of demand and will serve it later. As a consequence of this action, the penalty cost is performed.

Therefore, in comparison with other recourse models, see [14], we will solve the model for the expected demand case, and then, we will evaluate the consequence by computing a so called EEV characteristic. This will lead to a suboptimal solution in comparison with the typical two-stage recourse model by [14]. However, the model size does not grow exponentially with the increase of number of realizations of random demands. Thus, we do not need to deal with the unsolvable integer programs even for a relatively small data sets.

3.2. EEV (*Expected Effective Value*) for CVRPSD

In the CVRPSD, EEV is defined as $E_\xi\left(z(x_{min}^{EV}, d(\xi))\right)$ where z_{min}^{EV} denotes the objective function, E_ξ denotes the expected value and $d(\xi)$ is the stochastic demand. Here EEV measures the performance of x_{min}^{EV} for the underlying objective function, how good the solution x_{min}^{EV} to find the minimum objective function of CVRPSD problem.

The steps used in EEV can be described as follows:

1. Assume we have CVRP, then we solve the problem. For the problem with expected demands $E_{\xi}(d(\xi))$. We use the model of CVRP, as it is defined in chapter 2 section 2.4.
2. Let us take the value of x_{ij} that we have found from the first step and we insert it into the following model:

$$z = \sum_{i \in R}^{n} \sum_{i \neq j, j \in R}^{n} c_{ij} x_{ij} + \theta$$

Subject to:

$$y_{ij}^{s} + w_{ij}^{s} = d_{j}^{s}.x_{ij} \; \forall i, j \; and \; i \neq j$$

$$y_{ij}^{s}, w_{ij}^{s} \geq 0$$

where θ under recourse policy can be replaced by the following formula:

$$\theta = \sum_{s=1}^{S} p_{s}. \sum_{j \in R}^{n} \sum_{i \neq j, j \in R}^{n} l_{j}.y_{ij}^{s} + q_{j}.w_{ij}^{s}$$

Here, y_{ij}^{s} represent the amount of demand that can be served by vehicle at customer j, whereas w_{ij}^{s} represents the amount of demand that cannot be satisfied by vehicle. The index variable s represents the scenario and p_s is the probability of the scenario s that can appear. A parameter variable l_j represents the cost needed to distribute goods to customer j and q_j is a penalty cost at customer j as a consequence that the certain amount of demand at customer j cannot be serviced by vehicles.

In order to find the suitable set up value of l_j and q_j, observe the figure 3.1. Let us consider depot and node 3. It is easy to verify that the expected cost of l_3 is given by the edge between depot and customer 4. While the expected cost of q_3 can be determined by adding l_4 and $2l_4$ under assumption that the vehicle should return to depot when the demand

exceed the capacity. Therefore, the formula for finding the value of l_j and q_j can be generalized as follows:

$l_j = 1..n$

$q_j = l_j + 2l_j = 3c_{1j}, j = 1..n$

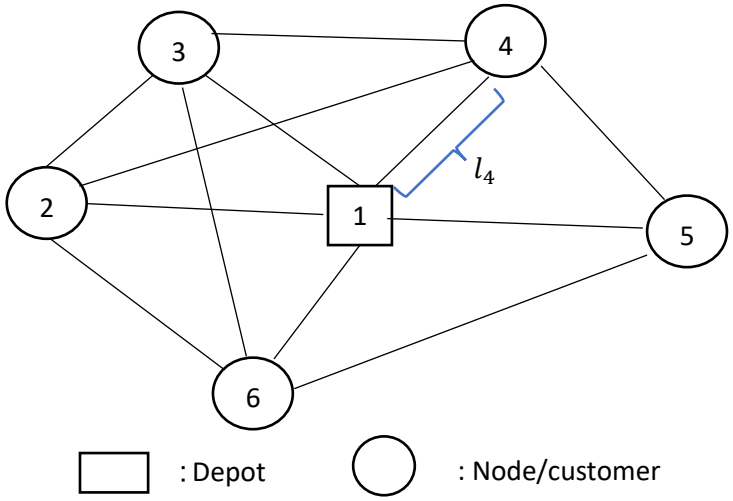

Figure 3.1: Illustration graph for finding the value of l_j and q_j

3. Solve the problem and we get the desire solution.

Based on the steps of EEV, we can formulate the CVRPSD by combining CVRP from chapter two and the recourse formulation stated as follows:

$$z = \sum_{i \in R}^{n} \sum_{i \neq j, j \in R}^{n} c_{ij} x_{ij} + \sum_{s=1}^{S} p_s . \sum_{j \in R}^{n} \sum_{i \neq j, j \in R}^{n} l_j . y_{ij}^{s} + q_j . w_{ij}^{s} \quad (17)$$

Subject to:

$$\sum_{j=2}^{n} x_{1j} = m$$

$$\sum_{j=2}^{n} x_{j1} = m$$

$$\sum_{i=1}^{n} x_{ij} = 1, j = 2,..n$$

$$\sum_{j=1}^{n} x_{ij} = 1, i = 2,..n$$

$$u_j - u_i + K(1 - x_{ij}) \geq d_j, i, j \in V\backslash\{1\}$$

$$y_{ij}^s + w_{ij}^s = d_j^s . x_{ij} \; \forall i, j \; and \; i \neq j$$

$$y_{ij}^s, w_{ij}^s \geq 0$$

$$x_{ij} \in \{0,1\}, \forall (i,j) \in A$$

In order to understand EEV approach, consider data from Table 3.1 and Table 3.2. In this case, there are only 3 scenarios. First, we need to define the probability distribution. Assume the probabilities are independent. To calculate the joint probabilities for all of demands, note that

$$p_1\big(d(n1) \wedge d(n2) \wedge d(n3) \wedge d(n4) \wedge d(n5) \wedge d(n6)\big) =$$

$$1 \times 0.3 \times 0.3 \times 0.3 \times 0.25 \times 0.3 = 0.002025$$

Then, by using formula (17), we obtain the cost for the problem is 163.36 with the routes and the solution are shown in Figure 3.2 and Figure 3.3.

Table 3.1. The coordinate of depot and customer for 5 customers

	x	y
(depot)	40	40
n(2)	22	22
n(3)	36	26
n(4)	21	45
n(5)	45	35
n(6)	55	20

Table 3.2. Demands and probability distribution of problem with 5 customers

	d(n(1))	d(n(2))	d(n(3))	d(n(4))	d(n(5))	d(n(6))
S(1)	0	19	20	16	23	11
Prob	1	0.3	0.3	0.3	0.25	0.3
S(2)	0	15	22	12	25	11
Pro	1	0.3	0.6	0.5	0.25	0.35
S(3)	0	17	24	14	27	11
Prob	1	0.4	0.1	0.2	0.5	0.35

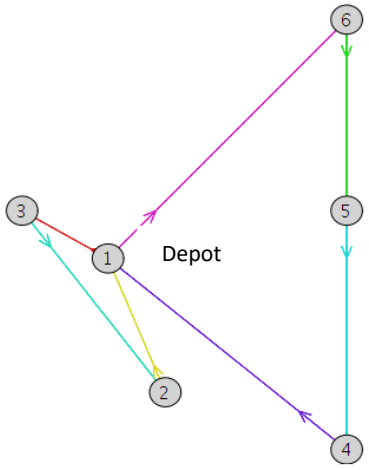

Figure 3.2: The routes of the problem of data from Table 3.1 and Table 3.2

$$Time\_execution := 0.281$$

$$Zp_s := [163.360776948583, [(w_{1,2})_1 = 0, (w_{1,2})_2 = 0, (w_{1,2})_3 = 0, (w_{1,3})_1 = 0, (w_{1,3})_2 = 0, (w_{1,3})_3 = 0, (w_{1,4})_1 = 0, (w_{1,4})_2 = 0, (w_{1,4})_3 = 0, (w_{1,5})_1 = 0, (w_{1,5})_2 = 0, (w_{1,5})_3 = 0, (w_{1,6})_1 = 0, (w_{1,6})_2 = 0, (w_{1,6})_3 = 0, (w_{2,1})_1 = 0, (w_{2,1})_2 = 0, (w_{2,1})_3 = 0, (w_{2,3})_1 = 0, (w_{2,3})_2 = 0, (w_{2,3})_3 = 0, (w_{2,4})_1 = 0, (w_{2,4})_2 = 0, (w_{2,4})_3 = 0, (w_{2,5})_1 = 0, (w_{2,5})_2 = 0, (w_{2,5})_3 = 0, (w_{2,6})_1 = 0, (w_{2,6})_2 = 0, (w_{2,6})_3 = 0, (w_{3,1})_1 = 0, (w_{3,1})_2 = 0, (w_{3,1})_3 = 0, (w_{3,2})_1 = 0, (w_{3,2})_2 = 0, (w_{3,2})_3 = 0, (w_{3,4})_1 = 0, (w_{3,4})_2 = 0, (w_{3,4})_3 = 0, (w_{3,5})_1 = 0, (w_{3,5})_2 = 0, (w_{3,5})_3 = 0, (w_{3,6})_1 = 0, (w_{3,6})_2 = 0, (w_{3,6})_3 = 0, (w_{4,1})_1 = 0, (w_{4,1})_2 = 0, (w_{4,1})_3 = 0, (w_{4,2})_1 = 0, (w_{4,2})_2 = 0, (w_{4,2})_3 = 0, (w_{4,3})_1 = 0, (w_{4,3})_2 = 0, (w_{4,3})_3 = 0, (w_{4,5})_1 = 0, (w_{4,5})_2 = 0, (w_{4,5})_3 = 0, (w_{4,6})_1 = 0, (w_{4,6})_2 = 0, (w_{4,6})_3 = 0, (w_{5,1})_1 = 0, (w_{5,1})_2 = 0, (w_{5,1})_3 = 0, (w_{5,2})_1 = 0, (w_{5,2})_2 = 0, (w_{5,2})_3 = 0, (w_{5,3})_1 = 0, (w_{5,3})_2 = 0, (w_{5,3})_3 = 0, (w_{5,4})_1 = 0, (w_{5,4})_2 = 0, (w_{5,4})_3 = 0, (w_{5,6})_1 = 0, (w_{5,6})_2 = 0, (w_{5,6})_3 = 0, (w_{6,1})_1 = 0, (w_{6,1})_2 = 0, (w_{6,1})_3 = 0, (w_{6,2})_1 = 0, (w_{6,2})_2 = 0, (w_{6,2})_3 = 0, (w_{6,3})_1 = 0, (w_{6,3})_2 = 0, (w_{6,3})_3 = 0, (w_{6,4})_1 = 0, (w_{6,4})_2 = 0, (w_{6,4})_3 = 0, (w_{6,5})_1 = 0, (w_{6,5})_2 = 0, (w_{6,5})_3 = 0, (y_{1,2})_1 = 0, (y_{1,2})_2 = 0, (y_{1,2})_3 = 0, (y_{1,3})_1 = 20, (y_{1,3})_2 = 20, (y_{1,3})_3 = 20, (y_{1,4})_1 = 0, (y_{1,4})_2 = 0, (y_{1,4})_3 = 0, (y_{1,5})_1 = 0, (y_{1,5})_2 = 0, (y_{1,5})_3 = 0, (y_{1,6})_1 = 10, (y_{1,6})_2 = 15, (y_{1,6})_3 = 19, (y_{2,1})_1 = 0, (y_{2,1})_2 = 0, (y_{2,1})_3 = 0, (y_{2,3})_1 = 0, (y_{2,3})_2 = 0, (y_{2,3})_3 = 0, (y_{2,4})_1 = 0, (y_{2,4})_2 = 0, (y_{2,4})_3 = 0, (y_{2,5})_1 = 0, (y_{2,5})_2 = 0, (y_{2,5})_3 = 0, (y_{2,6})_1 = 0, (y_{2,6})_2 = 0, (y_{2,6})_3 = 0, (y_{3,1})_1 = 0, (y_{3,1})_2 = 0, (y_{3,1})_3 = 0, (y_{3,2})_1 = 20, (y_{3,2})_2 = 10, (y_{3,2})_3 = 15, (y_{3,4})_1 = 0, (y_{3,4})_2 = 0, (y_{3,4})_3 = 0, (y_{3,5})_1 = 0, (y_{3,5})_2 = 0, (y_{3,5})_3 = 0, (y_{3,6})_1 = 0, (y_{3,6})_2 = 0, (y_{3,6})_3 = 0, (y_{4,1})_1 = 0, (y_{4,1})_2 = 0, (y_{4,1})_3 = 0, (y_{4,2})_1 = 0, (y_{4,2})_2 = 0, (y_{4,2})_3 = 0, (y_{4,3})_1 = 0, (y_{4,3})_2 = 0, (y_{4,3})_3 = 0, (y_{4,5})_1 = 0, (y_{4,5})_2 = 0, (y_{4,5})_3 = 0, (y_{4,6})_1 = 0, (y_{4,6})_2 = 0, (y_{4,6})_3 = 0, (y_{5,1})_1 = 0, (y_{5,1})_2 = 0, (y_{5,1})_3 = 0, (y_{5,2})_1 = 0, (y_{5,2})_2 = 0, (y_{5,2})_3 = 0, (y_{5,3})_1 = 0, (y_{5,3})_2 = 0, (y_{5,3})_3 = 0, (y_{5,4})_1 = 25, (y_{5,4})_2 = 15, (y_{5,4})_3 = 25, (y_{5,6})_1 = 0, (y_{5,6})_2 = 0, (y_{5,6})_3 = 0, (y_{6,1})_1 = 0, (y_{6,1})_2 = 0, (y_{6,1})_3 = 0, (y_{6,2})_1 = 0, (y_{6,2})_2 = 0, (y_{6,2})_3 = 0, (y_{6,3})_1 = 0, (y_{6,3})_2 = 0, (y_{6,3})_3 = 0, (y_{6,4})_1 = 0, (y_{6,4})_2 = 0, (y_{6,4})_3 = 0, (y_{6,5})_1 = 15, (y_{6,5})_2 = 20, (y_{6,5})_3 = 18]]$$

The Objective function is 163.360776948583

Figure 3.3: The solution of the problem of data from Table 3.1 and Table 3.2

3.3. Fuzzy C-Means (FCM) – TSP Method for finding Heuristic Solution of CVRPSD

Many construction heuristics for VRP fall into one of the three classes: insertion algorithm, savings algorithm and clustering algorithm. This research will focus on how to solve CVRPSD by using clustering. Clustering algorithms are two-phase algorithms. The first phase consists of grouping customers into clusters where each cluster should be served by one vehicle. The second phase then creates routes for each cluster.

In this research, Fuzzy C-Means (FCM) is chosen to handle the first phase since it is a powerful unsupervised method for the analysis of data and construction of models. This method was developed by Joe Dunn in 1973 and improved by Jim Bezdek in 1981 [17]. In many conditions, fuzzy clustering is more natural than hard clustering. Objects on the boundaries between several classes are not forced to fully belong to one of the classes, but rather are assigned membership degrees between 0 and 1 indicating their partial membership.

In general, the algorithm of FCM is described as follows:

Step 1 Initialize $U = [u_{ij}]$ matrix add iteration index 0, U^0, where u_{ij} is the degree of membership of variables x_i in the cluster j.

Step 2 At k-step, calculate the centers vectors $C^k = [c_j]$ with U^k, where

$$c_j = \frac{\sum_{i=1}^{n} u_{ij}^f . x_i}{\sum_{i=1}^{n} u_{ij}^f}$$

for f is any real number greater than 1 (fuzziness factor) and c_j is the cluster centers.

Step 3 Update U^k to obtain U^{k+1}

$$u_{ij} = \frac{1}{\sum_{k=1}^{C} \left(\frac{\left|\left|x_i - c_j\right|\right|}{\left|\left|x_i - c_k\right|\right|} \right)^{\frac{2}{f-1}}}$$

Step 4 If $\left|\left|U^{k+1} - U^{k}\right|\right| < \varepsilon$, then stop, otherwise return to step 2.

Figure 3.4 shows a schematic flowchart how the algorithm solves CVRPSD, where in the clustering part, system will cluster customers according to the coordinate of customers that will give value for variable $f = 2$.

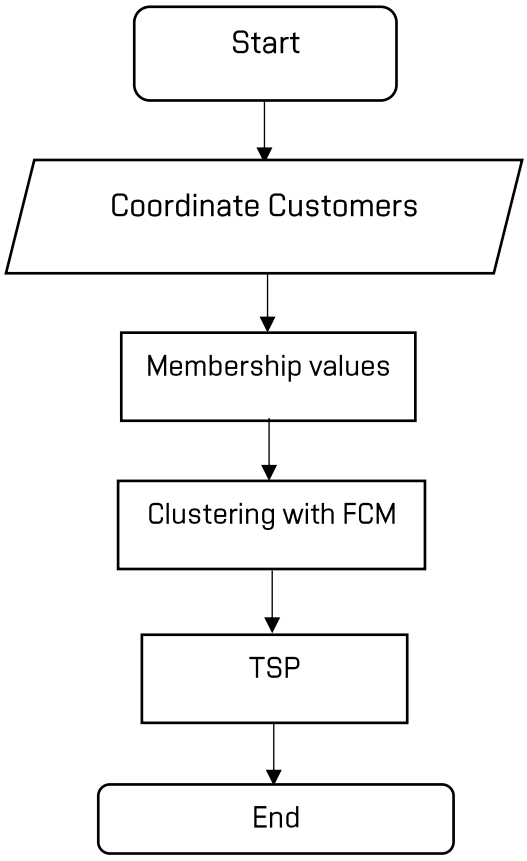

Figure 3.4. Flowchart of the method

For the second phase, we should create routes for each cluster, where in graph theory's point of view, this is *Travelling Salesman Problem with Stochastics Demand* (TSPSD) since we have to deal with stochastics demand. In order to solve this problem, let us state mathematical model of TSPSD in the following:

$$z = \sum_{e=1}^{m} \min_{e} \left\{ \sum_{i \in R}^{n} \sum_{i \neq j, j \in R}^{n} c_{ij} x_{ij} + \sum_{s=1}^{S} p_s . \sum_{j \in R}^{n} \sum_{i \neq j, j \in R}^{n} c_{1j} . y_{ij}^{s} + 3 c_{1j} . w_{ij}^{s} \right\}$$

Subject to:

$$\sum_{j \in R \setminus \{1\}}^{n} x_{1j} = 1$$

$$\sum_{j \in R \setminus \{1\}}^{n} x_{j1} = 1$$

$$\sum_{i=1}^{n} x_{ij} = 1, j = 2, .. n$$

$$\sum_{j=1}^{n} x_{ij} = 1, i = 2, .. n$$

$$u_i - u_j + n x_{ij} \leq n - 1, \quad 2 \leq i \neq j \leq n$$

$$x_{ij} \in \{0,1\}, \forall (i,j) \in A$$

$$y_{ij}^{s} + w_{ij}^{s} = d_j^{s} . x_{ij} \; \forall i, j \; and \; i \neq j$$

$$\sum_{j=1}^{n} \sum_{i=1, i \neq j}^{n} y_{ij}^{s} \leq K, j = 2, .. n$$

$$y_{ij}^{s}, w_{ij}^{s} \geq 0,$$

where R is a set of customers in a cluster and e is e^{th} route. Then, by using this model, we can get the desire solution.

Now consider the following example. Let us have given 4 identical vehicles of capacity 50. Suppose that the number of customers is 8. The coordinates of depot and customer can be seen in Table 3.2. So, the distances between depot and customer can be found by using Euclidean norm on R^2. Assume we have 3 scenarios for demand with the individual probability of every customer described in Table 3.2. Since there are 4 vehicles, we split customers into 4 clusters. Then, by

following the steps in Figure 3.1, the solution for this problem can be obtained. Here, the solution is represented both graphically and textually. Textual output contains a route number and a permutation of costumers (see in Table 3.3), whereas graphical output of the solution consists of routes, each having a different color and customers are illustrated as points in 2-D coordination system. For further, the graphical output can be seen in Figure 3.3.

Table 3.3. The Solution of example

Route	Sequence of route	Total Cost for each route	Recourse Action
1	1-2-3-1	54.96	No
2	1-4-8-1	58.45	No
3	1-5-6-7-1	66.95	Yes
4	1-9-1.	58.46	No

In route 3, we can see that recourse action is needed since the amount of actual demand exceeds the vehicle capacity may occur. For instance, in the scenario 2, the vehicle cannot be able to load all of demands at $n(5)$, i.e. the amount of demand can be served by vehicle only 14 of 25. For the remaining amount, we leave it and serve it later. As a consequence of this action, penalty cost is applied.

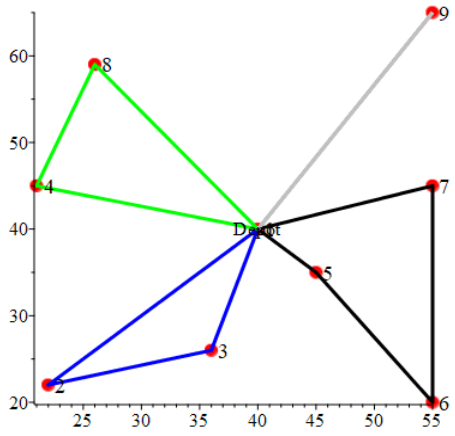

Figure 3.5. Graph representation for solution of the example

Table 3.4. The coordinate of depot and customer

	x	Y
(depot)	40	40
n(2)	22	22
n(3)	36	26
n(4)	21	45
n(5)	45	35
n(6)	55	20
n(7)	55	45
n(8)	26	59
n(9)	55	65

Table 3.5. Scenario's problem

	d(n(1))	d(n(2))	d(n(3))	d(n(4))	d(n(5))	d(n(6))	d(n(7))	d(n(8))	d(n(9))
S(1)	0	19	20	16	23	11	31	15	28
Prob	1	0.3	0.3	0.3	0.25	0.3	0.2	0.1	0.8
S(2)	0	15	22	12	25	11	25	18	20
Pro	1	0.3	0.6	0.5	0.25	0.35	0.6	0.45	0.1
S(3)	0	17	24	14	27	11	27	20	23
Prob	1	0.4	0.1	0.2	0.5	0.35	0.2	0.45	0.1

Chapter 4
Results

4.1 Computational Results and Discussion

In this section, we demonstrate the effectiveness and applicability of 2 algorithms (EEV and FCM-TSP) with best known results for CVRPSD and our test data sets. Algorithms were tested on 9 instances, 2 from random data (see on appendix A) and 7 instances taken from classical sets of the CVRP benchmark from Augerat et al. (set P) with some additional scenarios. The input data can be downloaded at [18]. Tests were ran on a PC with Windows 10 Home 64-bit operating system powered by Intel® Core™ i5 - 7200U, 2,5 GHz and 6 GB RAM where every instance is run 7 times. All algorithms were implemented in MAPLE 2018.

In order to prove the effectiveness of 2 algorithms, we consider 3 chosen instances (Random 1, Random 2, and P-n16-k8) when all of them have 3 scenarios. Then, we measure the performance of algorithms by calculating the relative percentage deviation (RPD) of the obtained best solution from the best-known solution (see definition 1). The results of calculation of RPD for both algorithms are given in Table 4.1.

Definition 1 *Let B be our best solution from 7 runs, B^* the best-known solution and C(B), $C(B^*)$ the cost of these solutions respectively. Then the relative percentage deviation (RPD) is calculated as follows:*

$$RPD = \frac{C(B) - C(B^*)}{C(B^*)}.100\%$$

Table 4.1. RPD results of chosen instances

Instances	K	n	m	Best Known	Method			
					EEV	RPD	FCM-TSP	RPD
Random 1	50	5	2	162.105	163.36	0.00773	231.614	0.428
Random 2	50	8	4	245.424	246.16	0.00299	238.842	-0.0268
P-n16-k8	35	15	8	450	451.33	0.00296	444.62	-0.0119

As can be seen in Table 4.1, the performance of EEV is better than FCM – TSP since $|\mathrm{RPD\ of\ EEV}| \leq |\mathrm{RPD\ of\ FCM-TSP}|$ for all of instances. This implies that the EEV will produce a better solution than FCM-TSP, where the solution is relatively close with the best-known solution. In the other hand, from table 4.1, we also found a fact that FCM-TSP in Random 1 case cannot perform well, while in the last 2 instances, FCM-TSP could produce the solutions that are lesser than best known results. That is why the value of their RPD are negative. However, in the future, the FCM-TSP solution can be afterwards evaluated by using all scenarios in the same way as in the EEV case.

Now, we take a look on how the EEV algorithm works for solving CVRPSD. Consider the instance Random 1 from Table 4.1. Here, assume that this is a CVRP problem that must be solved. It is easy to verify that the best-known solution for this instance is 162.105 with the set of certain value of x_{ij}. Then, let us take the values of x_{ij} from this problem to handle the case when the instance has 3 scenarios for demands. It gives the cost 163.36 with the computational time 0.281 second. For comparison (see Table 4.2 for more detail), in Random 2 with the same algorithm, it takes 9.922 seconds to get 246.16 as the cost. While in P-n16-k8, the computing time up to 25481.625 seconds ($\approx 7\ hours$) to get the cost 451.33. In general, it is not capable to solve a problem by EEV with more than 20 customers in an acceptable running time. In the case of utilizing the full scenario-based two-stage stochastic programming model designed by ideas of [14] it will even worse from the computational time point of view. Therefore, it explains our initial preference of EEV approach. Hence, we can still conclude that in the EEV, as the problem size increases, the computational time also increase exponentially. This

also gives another conclusion that because of time needs for finding an optimal (or enough good suboptimal) solution of problems on a big scale, the EEV is not implementable in the real situation that usually involves a large amount of data. Therefore, a heuristics approach is introduced as a shortcut for solving big scale problem.

Table 4.2. The result of EEV of some instances

Instances	K	n	m	Solution of EEV	
				n(s)=3	time(second)
Random 1	50	5	2	163.36	0.281
Random 2	50	8	4	246.16	9.922
P-n16-k8	35	15	8	451.33	25481.625

In the heuristic approach, it is not always necessary to get the best solution, but the good one is just as useful if the result is approximately as close as possible with the exact solution and it can be found in a short time. In the thesis, the FCM-TSP (see Section 3.3) is chosen to find the heuristic solution of some problems. As we have known from previous chapter, FCM-TSP is an algorithm that consist of two phases. In the first phase, grouping customers into clusters should be done. Then, for the second phase, we should create route for each cluster such that every cluster is served by one vehicle. Table 4.3 represents a heuristics solution of 9 instances when we have 3 scenarios. Based on Table 4.3, the general conclusion in the routing problem saying “when the number of customer increases, an exponential rise of time in the computations is required” is not totally true (at least for our tests). For instance, in P-n50-k7, the solution can be computed in a time around 2007.015 seconds, whereas in P-n55-k15, system only need 109.266 seconds to solve the problem.

In order to identify this idea, observe the solution of these 2 instances in Figure 4.1, Table 4.4 and Table 4.5. Overall, the average of number of customers of each cluster in P-n50-k7 is greater than P-n55-k15. At the same time, we can see that FCM-TSP could solve P-n55-k15 faster than P-n50-k7. So, we can conclude

for these specific test computations that the speed of FCM-TSP solution for finding heuristics based suboptimal solution does not depend on the total customers, but it depends on the number of customers in a cluster. Although the number of total customers is big enough, as long as the number of customers on each cluster is small, system does not need longer time for finding the solution.

Table 4.3. The result of some instances for heuristics solution

Instances	K	n	m	Solution	
				n(s)=3	time(second)
Random 1	50	5	2	231.614	3.434
Random 2	50	8	4	238.842	4.781
P-n16-k8	35	15	8	444.62	13.266
P-n22-k8	3000	21	8	582.86	12.063
P-n45-k5	150	44	5	524.074	25689.282
P-n50-k7	150	49	7	612.17	2007.015
P-n51-k10	80	50	10	763.896	120.719
P-n55-k15	70	54	15	932.223	109.266
P-n60-k15	150	59	15	923.583	196.985

Table 4.4. The solution of P-n55-k15

Route	Sequence of route	Total Cost for each route
1	1-18-4-45-41-13-1	49.5
2	1-32-11-39-1	84.48
3	1-34-2-3-1	54.98
4	1-29-22-37-48-49-1	77.78
5	1-40-10-33-1	55.89
6	1-36-15-12-54-1	69.22
7	1-46-30-31-1	41.93
8	1-16-21-38-6-1	76.12
9	1-7-17-52-1	40.97
10	1-23-43-42-44-1	82.06
11	1-20-55-14-1	62.38
12	1-27-1.	12.16
13	1-8-9-47-35-53-28-5-1	54.16
14	1-25-50-24-1	81.64
15	1-26-19-51-1	80.23

Table 4.5. The solution of P-n50-k7

Route	Sequence of route	Total Cost for each route
1	1-17-50-25-19-45-4-1	85.92
2	1-18-41-33-26-10-40-13-1	72.19
3	1-5-28-46-30-49-31-29-3-7-1	82.09
4	1-6-16-21-38-37-48-22-1	97.74
5	1-24-44-42-43-23-2-34-1	97.15
6	1-27-8-36-15-20-14-9-47-35-1	82.27
7	1-32-11-39-12-1	94.77

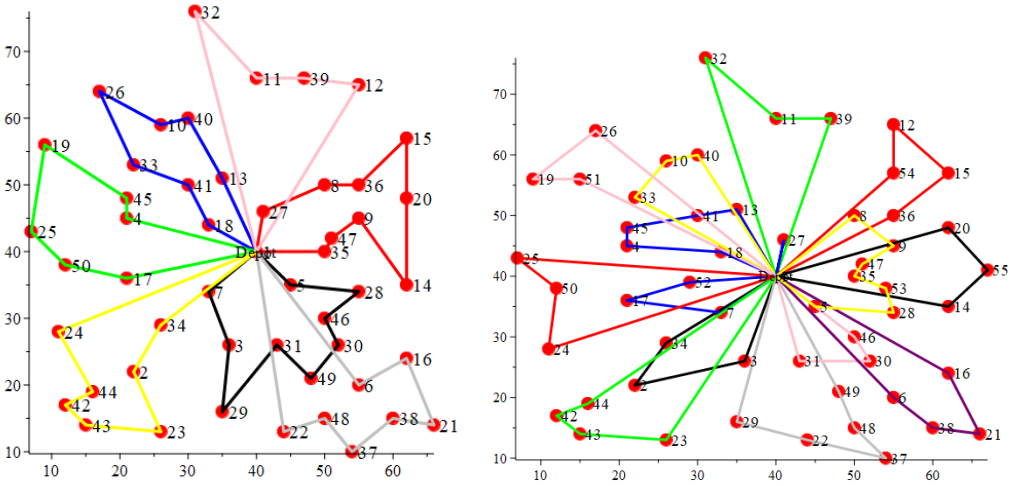

Figure 4.1. The solution for instance P-n50-k7 (left) and P-n55-k15(right)

Now, consider the case when we enlarge the problems by increasing the number of scenarios (i.e. $n(S) = 32$ and 162). As noted in Table 4.6, the action of adding the number of scenarios will increase the computational time, but there are no dramatic differences in the results of all of instances (as it is shown on Figure 4.2). Hence the number of scenarios does not affect the solution too much.

Table 4.6. The result of some instances after adding some scenarios

Instances	best known	Heuristics Solution			
		n(s)=32	time(second)	n(s)=162	time(second)
Random 1	162.105	226.698	3.218	192.844	3.453
Random 2	245.424	237.447	3.531	236.909	3.907
P-n16-k8	450	444.66	10.39	444.62	16.183
P-n22-k8	603	597.271	10.797	597.27	11.5

From the computational results and conducting analysis on them, it is observable that the EEV has given a better performance than FCM-TSP for solving CVRPSD in small instances. But the EEV has disadvantage, that the EEV is not capable to solve big instances in an acceptable running time because of complexity of the problems. In the other hand, the FCM-TSP is more applicable in the real situation because it is faster than the EEV. Moreover, the FCM-TSP also could solve big instances in a short time.

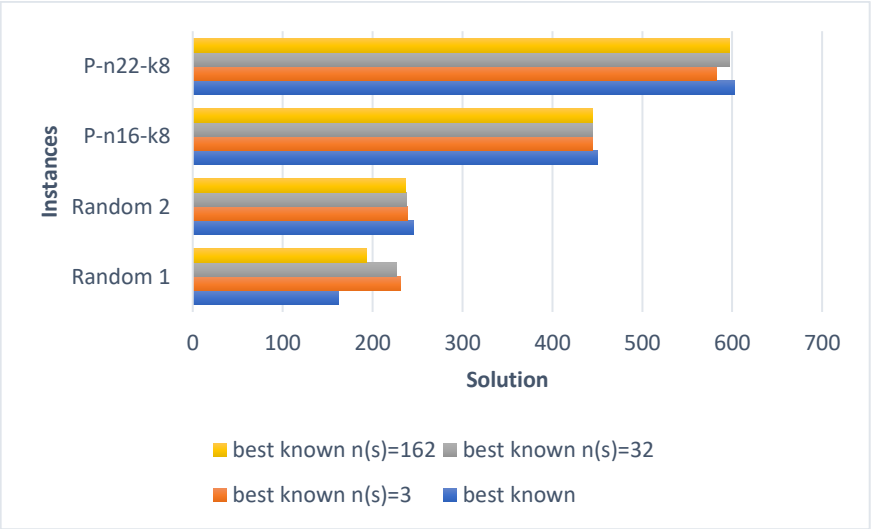

Figure 4.2. The solution of some instances depending the number of scenarios

Chapter 5
Conclusion

We have shown how to solve the vehicle routing problem with stochastic demands, which is an important idea in the real situation. From the results obtained, we found that FCM – TSP can be an alternative way to solve CVRPSD, rather than using the EEV approach since the EEV is only effective in small instances (i.e. instances involving less than 16 vertices). Moreover, FCM-TSP is faster than EEV to find the solution. The FCM – TSP approach can decrease the usage of computer memory to further finding of the optimal routes since the algorithm divides the customers into *n* clusters that can decrease the complexity of problem. But, the FCM – TSP has a weakness that is the computational time is depend on the number of customers in a cluster. In addition, results have confirmed that our modification by adding the number of scenarios will increase the computational time, but it does not affect the solution too much.

For future research efforts one should concentrate on gradually relaxing some of the underlying assumptions of the model and solve it by the other algorithm which is better than the FCM – TSP. In opposite to the further development of heuristics, some application of decomposition based algorithms (as e.g., Benders approach) should be discussed, as well.

References

[1] Z. Borcinova. Two models of the capacitated vehicle routing problem. *Croatian Operational Research Review,* no. 8, pp. 463-469, 2017.

[2] N. D. Jaegere, M. Defraeye and I. V. Nieuwenhuyse. The Vehicle Routing Problem: State of the Art Classification and Review. 2015.

[3] S. Ropke. Heuristic and exact algorithms for vehicle routing problems. Copenhagen, DTU library, 2006.

[4] N. Christofides. An Algorithmic Approach. Academic Press 1975.

[5] I. K. Budayasa. Teori Graph dan Aplikasinya. Surabaya: Unesa University Press, 2007.

[6] G. Ghiani, G. Laporte and R. Musmanno. Introduction to logistics systems planning and control. New Jersey: J. Wiley, 2004.

[7] G. Pataki. The bad an The good and ugly formulations for travelling salesman problem. Columbia University, Columbia, 2000.

[8] P. Toth and D. Vigo. Vehicle Routing: Problems, Meth-ods, and Applications, Second Edition. Bologna, 2014.

[9] M. Macho, Vehicle routing problem with time windows and its solving methods. Prague, 2016.

[10] T. Bektas. The multiple traveling salesman problem: an overview of formulations and solution procedures. vol. 34, 2006.

[11] T. Caric, A. Galic, J. Fosin, H. Gold and A. Reinholz. A Modelling and Optimization Framework for Real-World Vehicle Routing Problems. Dortmund.

[12] P. Raden. Aplikasi Kombinatorial pada Vehicle Routing Problem. Surabaya, 2007.

[13] B. F. Moghadam, S. J. Sadjadi and S. M. Seyedhosseini. Comparing Mathematical and Heuristic Methods for Robust Vehicle Routing Problem. *IJRRAS,* vol. II, no. 2, pp. 108-116, 2010.

[14] J. R. Birge and F. Louveaux. Introduction to Stochastic Programming Second Edition. New York: Springer, 2011.

[15] G. Laporte, F. V. Louveaux and L. V. Hamme. An Integer L-Shaped Algorithm for the Capacitated Vehicle Routing Problem with Stochastic Demands. *Operation Research,* pp. 415-423, 2002.

[16] C. H. Christiansen and J. Lysgaard. A column Generation Approach to the Capacitated Vehicle Routing Problem. Aarhus, 2006.

[17] R. Suganya and R. Shanthi. Fuzzy C-Means Algorithm- A Review. *International Journal of Scientific and Research Publications,* vol. II, no. 11, 2012.

[18] "CVRP-LIB," [Online]. Available: http://vrp.atd-lab.inf.puc-rio.br/index.php/en/. [Accessed 23 February 2019].

List of Figures

Figure 2.1. Graph $G1$ 11
Figure 2.2. Graph $G2$ 11
Figure 2.3. Connected graph $G3$ and disconnected graph $G4$.....13
Figure 2.4. Weighted graph.....13
Figure 2.5. TSP on a graph with weights on its edges.....15
Figure 2.6. VRP Variants.....17
Figure 3.1. Illustration graph for finding the value of l_j and q_j 24
Figure 3.2. The routes of the problem of data from Table 3.1 and Table 3.2 24
Figure 3.3. The solution of the problem of data from Table 3.1 and Table 3.2.....24
Figure 3.4. Flowchart of the method 26
Figure 3.5. Graph representation for solution of the example 28
Figure 4.1. The solution for instance P-n50-k7 (left) and P-n54-k15(right).....34
Figure 4.2. The solution of some instances depending the number of scenarios .35
Figure B-1. routes for 5 Customers 44
Figure B-2. routes for 8 Customers 44
Figure B-3. routes for P-n16-K8.....44
Figure B-4. routes for P-n22-K8.....44
Figure B-5. routes for P-n45-K5.....44
Figure B-6. routes for P-n50-K7.....44
Figure B-7. routes for P-n51-K10.....45
Figure B-8. routes for P-n55-K15.....45
Figure B-9. routes for P-n60-K15.....45

List of Tables

Table 2.1. Some methods to solve VRP 16
Table 3.1. The coordinate of depot and customer for 5 customers 29
Table 3.2. Demands and probability distribution of problem with 5 customers ..29
Table 3.3. The Solution of example.....29
Table 3.4. The coordinate of depot and customer 29
Table 3.5. Scenario's problem.....29
Table 4.1. RPD results of chosen instances.....31
Table 4.2. The result of EEV of some instances 32
Table 4.3. The result of some instances for heuristics solution 33
Table 4.4. The solution of P-n55-k15 33
Table 4.5. The solution of P-n50-k7 34
Table 4.6. The result of some instances after adding some scenarios.....34
Table A-1. The coordinate of depot and customer for 5 customers 42
Table A-2. Scenario's problem of 5 customers 42
Table A-3. The coordinate of depot and customer for 8 customers 42
Table A-4. Scenario's problem of 8 customers 43

Abbreviation list

TSP	Travelling Salesman Problem
VRP	Vehicle Routing Problem
CVRP	Capacitated – Vehicle Routing Problem
CVRPSD	Capacitated – Vehicle Routing Problem with Stochastic Demands
EEV	Expected Effective Value
FCM	Fuzzy C-Means

Appendix A

Table A-1. The coordinate of depot and customer for 5 customers

	x	y
(depot)	40	40
n(2)	22	22
n(3)	36	26
n(4)	21	45
n(5)	45	35
n(6)	55	20

Table A-2. Scenario's problem of 5 customers

	d(n(1))	d(n(2))	d(n(3))	d(n(4))	d(n(5))	d(n(6))
S(1)	0	19	20	16	23	11
Prob	1	0.3	0.3	0.3	0.25	0.3
S(2)	0	15	22	12	25	11
Pro	1	0.3	0.6	0.5	0.25	0.35
S(3)	0	17	24	14	27	11
Prob	1	0.4	0.1	0.2	0.5	0.35

Table A-3. The coordinate of depot and customer for 8 customers

	x	y
(depot)	40	40
n(2)	22	22
n(3)	36	26
n(4)	21	45
n(5)	45	35
n(6)	55	20
n(7)	55	45
n(8)	26	59
n(9)	55	65

Table A-4. Scenario's problem of 8 customers

	d(n(1))	d(n(2))	d(n(3))	d(n(4))	d(n(5))	d(n(6))	d(n(7))	d(n(8))	d(n(9))
S(1)	0	19	20	16	23	11	31	15	28
Prob	1	0.3	0.3	0.3	0.25	0.3	0.2	0.1	0.8
S(2)	0	15	22	12	25	11	25	18	20
Pro	1	0.3	0.6	0.5	0.25	0.35	0.6	0.45	0.1
S(3)	0	17	24	14	27	11	27	20	23
Prob	1	0.4	0.1	0.2	0.5	0.35	0.2	0.45	0.1

Appendix B

The Heuristic Solution of 8 instances

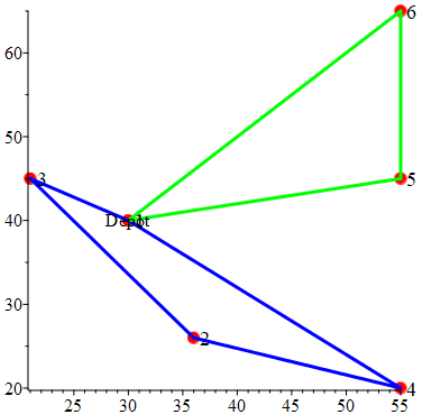

Figure B-1. routes for 5

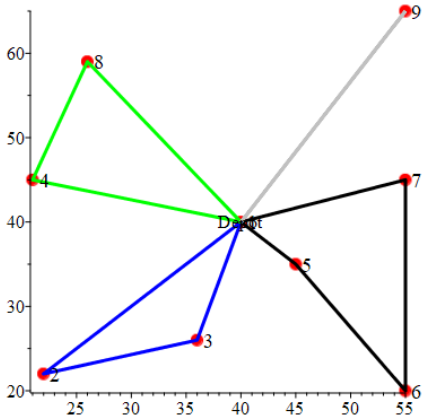

Figure B-2. routes for 8 Customers

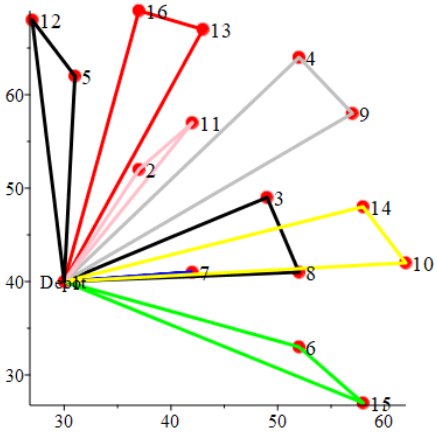

Figure B-3. routes for P-n16-K8

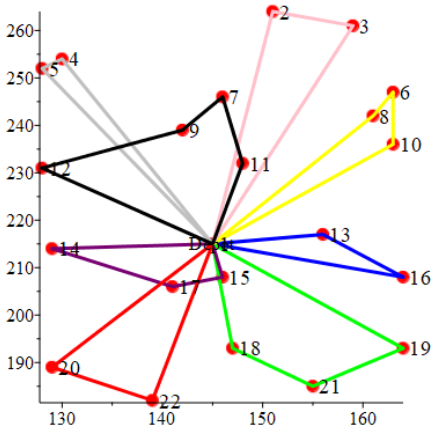

Figure B-4. routes for P-n22-K8

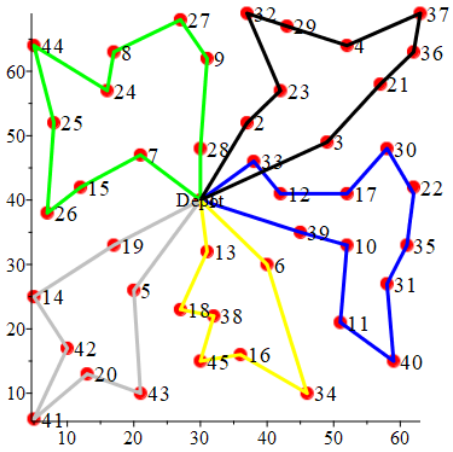

Figure B-5. routes for P-n44-K5

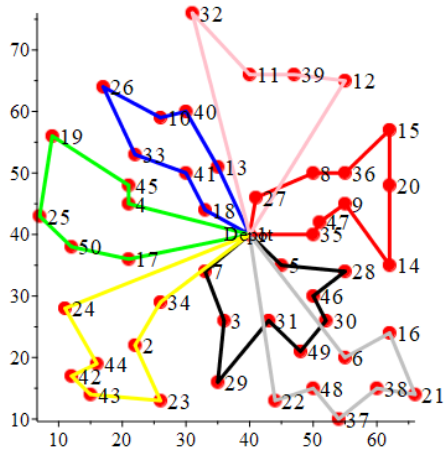

Figure B-6. routes for P-n50-K7

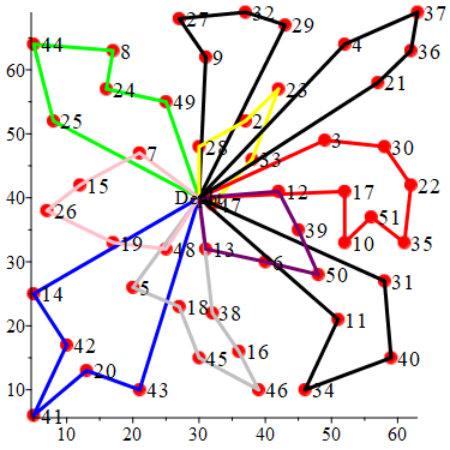

Figure B-7. routes for P-n51-K10

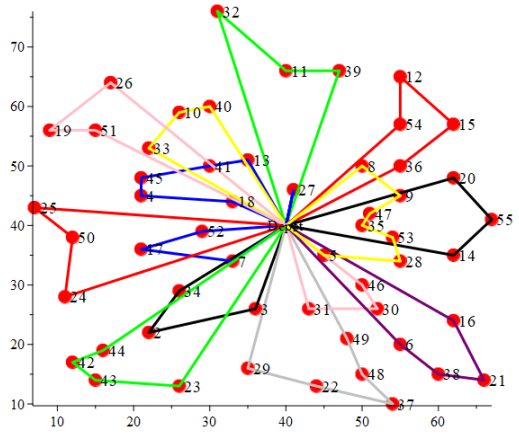

Figure B-8. routes for P-n55-K15

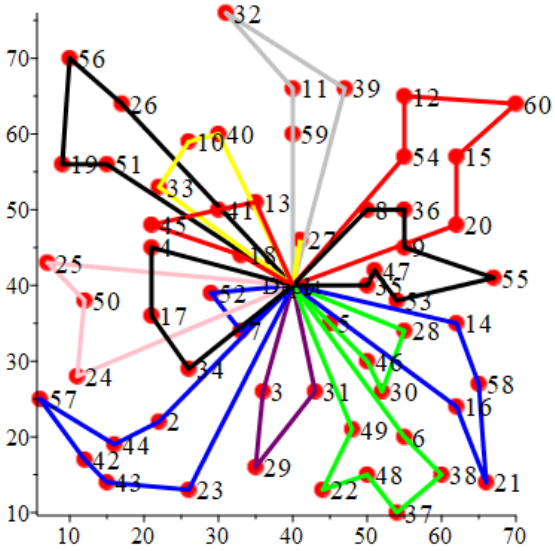

Figure B-9. routes for P-n60-K15

Appendix C

C.1. Sample Code for EEV

restart:
$gc();\ stgc := kernelopts(gctimes);$
$st := time();$
$city$
$= [[40,40],[22,22],[36,26],[21,45],[45,35],[55,20],[55,45],[26,59],[55,65]];$
$node := nops(city);$
$N := \{seq(1..node)\};$
$S := \{seq(1..3)\};$
$c := array(1..node, 1..node, [evalf(seq([seq(((city[i][1] - city[j][1])\text{^}2$
$+ (city[i][2] - city[j][2])\text{^}2)\text{^}(1/2), j = N)], i = N))]);$
$K := 50;\ m := 4;$
$d := \text{`}<,>\text{`}(0,19,20,16,23,11,31,15,28);$

$z := add(add(c[i,j] * x[i,j], i = N\ minus\ \{j\}), j = N);$
$con2 := seq(add(x[i,j], j = N\ minus\ \{i\}) = m, i = 1);$
$con3 := seq(add(x[j,i], j = N\ minus\ \{i\}) = m, i = 1);$
$con4 := seq(add(x[i,j], i = N\ minus\ \{j\}) = 1, j = N\ minus\ \{1\});$
$con5 := seq(add(x[j,i], i = N\ minus\ \{j\}) = 1, j = N\ minus\ \{1\});$
$con6 := seq(seq(u[i] - u[j] + K * (1 - x[i,j]) >= d[j], i$
$= N\ minus\ \{1,j\}), j = N\ minus\ \{1\});$
$route :$
$= Optimization[LPSolve](z, \{con2, con3, con4, con5, con6\}, binaryvariables$
$= \{seq(seq(x[i,j], i = N\ minus\ \{j\}), j = N)\});$
$x_l := [convert$
$(select(proc\ (i)options\ operator, arrow;\ has(i,x)end\ proc, [route[2][]]), set)[]];$
$Z0 := route[][1];$
$X := eval(Matrix(node, symbol = x), \{route[2][], seq(x[i,i] = 0, i$
$= 1..node)\});$
$settime := time();\ expression;\ time() - settime;\ gct :$
$= kernelopts(gctimes);$
$if\ gct <> stgc\ then\ gc()end\ if;$
$Time\_execution := time() - st;$

 # EEV Model
$d[1] := \text{`}<,>\text{`}(0,19,20,16,23,11,31,15,28);$
$d[2] := \text{`}<,>\text{`}(0,15,22,12,25,11,25,18,20);$
$d[3] := \text{`}<,>\text{`}(0,17,24,14,27,11,27,20,23);$
$probability[1] := \text{`}<,>\text{`}(1,.3,.3,.3,.25,.3,.2,.1,.8);$

```
probability[2] := `<,>`(1,.3,.6,.5,.25,.35,.6,.45,.10);
probability[3] := `<,>`(1,.4,.1,.2,.5,.35,.2,.45,.1);
for i to node do dist_cost[i] := c[1,i] end do;

l := `<,>`(seq(dist_cost[i],i = 1..node));
q := `<,>`(seq(3*dist_cost[i],i = 1..node));
z := add(add(c[i,j]*X[i,j],i = N minus {j}),j
        = N)+add(mul(probability[s])*add(add(l[j].y[i,j][s]
        +q[j]*w[i,j][s],i = N minus {j}),j = N),s = S);
constr1 := seq(seq(seq(y[i,j][s]+w[i,j][s] = d[s][j]*X[i,j],i
        = N minus {j}),j = N),s = S); constr3 :
        = seq(seq(seq(y[i,j][s] >= 0,i = N minus {j}),j = N),s
        = S);
constr4 := seq(seq(seq(w[i,j][s] >= 0,i = N minus {j}),j = N),s
        = S);
Zp[s] := Optimization[LPSolve](z,{constr1,constr3,constr4},assume
        = nonnegative*integer); Z_opt := Zp[s][1];
Typesetting:-mrow(Typesetting:-mn("The Objective function is ",color
= black,Style
= Bold),Typesetting:-Typeset(Typesetting:-EV(Z_opt)));
```

#Draw Graph Solution

```
GT := GraphTheory;
G := GT:-Graph(X);
GT:-SetVertexPositions(G,city);
GT:-DrawGraph(G);
Dist := proc (v1,v2) options operator,arrow; evalf(norm(`<,
        >`(`~`[`-`](city[v1],city[v2])),2)) end proc;
E := [GT:-Edges(G)[]];
dists := `~`[Dist@op](E); maxD,minD := (max,min)(dists);
GT:-HighlightEdges(G,E,[seq(COLOR(HSV,(.85*(d-minD))/(maxD
        -minD),.85,.85),d = dists)]);
GT:-DrawGraph(G);
```

C.2. Sample Code for FCM-TSP

```
restart:
# Clustering
gc(); stgc := kernelopts(gctimes);
st := time(); FuzzyCMeans := proc (X :: Matrix, {clusters ::
(And(posint, satisfies(proc (x) options operator, arrow; 1
                < x end proc))) := 8, fuzziness
                :: (satisfies(proc (x) options operator, arrow; 1
                < x end proc)) := 2, epsilon :: positive :
                = 0.1e - 3, max_iters :: posint := 999, Norm :: procedure :
                = proc (x) options operator, arrow;
LinearAlgebra: -Norm(x, 2) end proc }) local c, m, n, d, N, U, U1, C, e, i, j, k, jj, UM;
option testing; description Fuzzy C-Means clustering algorithm. ; c :
                = clusters; m := fuzziness; n := op([1, 1], X);
d := op([1, 2], X); N := Norm;
U := Matrix(n, c, datatype = float[8]);
U1 := LinearAlgebra: -RandomMatrix(n, c, generator
                = 0 .. 1, datatype = float[8]); C :
                = Matrix(c, d, datatype = float[8]);
e := 2/(m - 1);
for jj to max(iters) while epsilon
                <= LinearAlgebra: -Norm(U - U1, infinity) do
    U := copy(U1); UM := `~`[`^`](U, m); for j to c do C[j, () .. ()] :
                = add(UM[i, j] .X[i, () .. ()], i = 1 .. n)/add(UM[i, j], i
                = 1 .. n) end do; U1 :
                = Matrix(n, c, proc (i, j) options operator, arrow; 1
                /add((N(X[i, () .. ()] - C[j, () .. ()])/N(X[i, () .. ()]
                - C[k, () .. ()]))^e, k = 1 .. c) end proc, datatype
                = float[8]) end do; if max_iters
                < jj then error "Didn't converge" end if;
    userinfo(1, FuzzyCMeans, "Used", jj, "iterations."); C, U1 end proc
X :
= [[151, 264], [159, 261], [130, 254], [128, 252], [163, 247], [146, 246], [161, 242],
[142, 239], [163, 236], [148, 232], [128, 231], [156, 217], [129, 214], [146, 208],
 [164, 208], [141, 206], [147, 193], [164, 193], [129, 189], [155, 185], [139, 182]];
 XM := Matrix(X, datatype = float[8]); infolevel[FuzzyCMeans] := 1;
C, U := FuzzyCMeans(XM, clusters = 8, fuzziness = 2, epsilon
= 0.1e - 3, max_iters = 999, Norm
= (proc (x) options operator, arrow; LinearAlgebra: -Norm(x, 2) end proc));
```

```
AssignToClusters := proc (U
                :: Matrix) options operator, arrow; map(proc (i) local p;
member(max(U[i, () .. ()]), U[i, () .. ()], p); p end proc, [`$`(1 .. op([1, 1], U))])
                    end proc;
                    _local(A, k, J, c); 1;
                    A := AssignToClusters(U); 1;

J := [`$`(1 .. op([1, 1], U))]; 1; c := op([1, 2], U); a := [0, op(A)]; K :
                = 3000;
city := [[145, 215], [151, 264], [159, 261], [130, 254], [128, 252],
[163, 247], [146, 246], [161, 242], [142, 239], [163, 236], [148, 232], [128, 231],
[156, 217], [129, 214], [146, 208], [164, 208], [141, 206], [147, 193], [164, 193],
[129, 189], [155, 185], [139, 182]];
N := nops(city);
Dist := array(1 .. N, 1 .. N, [evalf(seq([seq(((city[i][1] - city[j][1])^2
                + (city[i][2] - city[j][2])^2)^(1/2), j = 1 .. N)], i
                = 1 .. N))])
```

 # Scenario

```
nod[1] := [0]; nod[2] := [1100, 1300, 1500]; nod[3] := [700]; nod[4] :
                = [800]; nod[5] := [1400]; nod[6] :
                = [2100, 1100, 1700]; nod[7] := [400]; nod[8] :
                = [800]; nod[9] := [100]; nod[10] := [500]; nod[11] :
                = [600, 1200, 1500]; nod[12] := [1200]; nod[13] :
                = [1300, 2000, 2200]; nod[14] := [1300]; nod[15] :
                = [300]; nod[16] := [900]; nod[17] := [2100]; nod[18] :
                = [1000, 900]; nod[19] := [900]; nod[20] :
                = [2500]; nod[21] := [1800]; nod[22] := [700];
probability[1] := [1]; probability[2] := [.5]; probability[3] :
                = [.5, .25, .25]; probability[4] := [.5]; probability[5] :
                = [.5]; probability[6] := [.6, .2, .2]; probability[7] :
                = [.4]; probability[8] := [.5]; probability[9] :
                = [.5]; probability[10] := [.5]; probability[11] :
                = [.5, .3, .2]; probability[12] := [.5]; probability[13] :
                = [.1]; probability[14] :
                = [.9, 0.5e - 1, 0.5e - 1]; probability[15] :
                = [.59]; probability[16] := [.41]; probability[17] :
                = [.2]; probability[18] := [.3]; probability[19] :
                = [.5, .5]; probability[20] := [.3]; probability[21] :
                = [.4]; probability[22] := [.1];
for i to N do dist_cost[i] := Dist[1, i] end do;
```

```
d :
= [seq(seq(seq(seq(seq(seq(seq(seq(seq(seq(seq(seq(seq(seq(seq(seq
(seq(seq(seq(seq(seq(` <, > `(a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v), a
                = nod[1]), b = nod[2]), c = nod[3]), d = nod[4]), e
                = nod[5]), f = nod[6]), g = nod[7]), h = nod[8]), i
                = nod[9]), j = nod[10]), k = nod[11]), l = nod[12]), m
                = nod[13]), n = nod[14]), o = nod[15]), p = nod[16]), q
                = nod[17]), r = nod[18]), s = nod[19]), t = nod[20]), u
                = nod[21]), v = nod[22])];
S := numelems(d);
pro_scenario :
= [seq(seq(seq(seq(seq(seq(seq(seq(seq(seq(seq(seq(seq(seq(seq(seq
(seq(seq(seq(seq(seq(` <, > `(a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v), a
                = probability[1]), b = probability[2]), c = probability[3]), d
                = probability[4]), e = probability[5]), f = probability[6]), g
                = probability[7]), h = probability[8]), i = probability[9]), j
                = probability[10]), k = probability[11]), l
                = probability[12]), m = probability[13]), n
                = probability[14]), o = probability[15]), p
                = probability[16]), q = probability[17]), r
                = probability[18]), s = probability[19]), t
                = probability[20]), u = probability[21]), v
                = probability[22])];
```

 # TSPSD

```
for Ni to c do Typesetting: −mrow(Typesetting: −mn
("MODEL SYSTEM FOR ROUTE ", color = black, Style
                = Bold), Typesetting: −Typeset(Typesetting: −EV(Ni)));
h := convert(select(proc (i) options operator, arrow;
a[i] = Ni end proc, [`$`(1 .. nops(a))]), set); df :
                = select(proc (i) options operator, arrow;
a[i] = Ni end proc, [`$`(1 .. nops(a))]);
dt := subsop(nops(df) = NULL, df);
r := h union {1}; n := numelems(h);
l[Ni] := ` <, > `(seq(dist_cost[i], i = r));
q[Ni] := ` <, > `(seq(3 ∗ dist_cost[i], i = r));
z[Ni] := add(add(Dist[i, j] ∗ x[i, j], i = r minus {j}), j
                = r) + add(mul(pro_scenario[s]) ∗ add(add(add(l[Ni][cf]
                ∗ y[s][i, j] + q[Ni][cf] ∗ w[s][i, j], j = r minus {i}), cf
                = 1 .. numelems(r)), i = r), s = S);
conv1 := seq(add(x[i, j], j = r minus {i}) = 1, i = r);
conv2 := seq(add(x[j, i], j = r minus {i}) = 1, i = r);
conx1 := seq(add(x[i, j], i = r minus {j}) = 1, j = r minus {1});
```

```
conx2 := seq(add(x[j, i], i = r minus {j}) = 1, j = r minus {1});
conu := seq(seq(n * x[i, j] + u[i] - u[j] <= n - 1, i = r minus {1, j}), j
        = r minus {1});
constr1 := seq(seq(seq(y[s][i, j] + w[s][i, j] = d[s][j] * x[i, j], j
        = r minus {i}), i = r), s = S); constr2 :
        = seq(add(add(y[s][i, j], j = r minus {i}), i = r) <= K, s
        = S);
constr3 := seq(seq(seq(y[s][i, j] >= 0, i = r minus {j}), j = r), s = S);
constr4 := seq(seq(seq(w[s][i, j] >= 0, i = r minus {j}), j = r), s = S);
route[Ni] :
= Optimization[LPSolve](z[Ni], {constr1, constr2, constr3, constr4, conu, conv1,
conv2, conx1, conx2}, depthlimit = 2000, integervariables
        = [seq(seq(seq(y[s][i, j], i = r minus {j}), j = r), s
        = S), seq(seq(seq(w[s][i, j], i = r minus {j}), j = r), s
        = S)], binaryvariables = {seq(seq(x[i, j], i = r minus {j}), j
        = r)}); Z0[Ni] := route[Ni][][1] end do; Z_opt := add(Z0[i], i
        = 1 .. c);
Typesetting: -mrow(Typesetting: -mn("The Objective function is ", color
        = black, Style
        = Bold), Typesetting: -Typeset(Typesetting: -EV(Z_opt)));
settime := time(); expression; time() - settime;
gct := kernelopts(gctimes);
                if gct <> stgc then gc() end if


Time_execution := time() - st;
```